A Software Development Process for COTS-Based Information System Infrastructure

Part II: Lessons Learned

Greg Fox and Steven Marcom, *TRW*
Karen W. Lantner, *EDS*

Part I of this article (CROSSTALK, March 1998) described the Infrastructure Incremental Development Approach process model. Part II describes a particular application of that model and examines the practical lessons learned and pitfalls encountered.

Modern software developers are guided by a variety of formal and informal processes that help to organize and control development activities across large groups of developers or multiple organizations. These processes supply the discipline and order lacking in many early development efforts. The currently available inventory of documented process methods has a limitation: most assume the system being built will be coded largely from scratch. As a result, the processes do not address many of the challenges associated with building systems that contain large amounts of commercial-off-the-shelf (COTS) software.

The Infrastructure Incremental Development Approach (IIDA) is a combination of the classical development model and the spiral process model to accommodate the needs of COTS-based technical infrastructure development. Each stage of the development cycle is augmented with a series of structured prototypes for COTS product evaluation and integration. This close coupling of prototyping and development stages characterizes the IIDA. The critical success factors for this method are the early establishment of an integrated development environment in which to install the COTS products and early planning for the prototype integration and testing environment, including simulated applications and other test software.

Application of IIDA

The following describes experiences using the IIDA method from 1994 to 1997 to develop the initial versions of an infrastructure to support business applications developers for a large, enterprise-wide heterogenous system. The types of COTS products that were integrated included

- Operating systems provided by four different vendors.
- End-user interface COTS software to provide a common graphical user interface.
- Middleware COTS products to provide a uniform transaction processing capability.
- Combinations of COTS software and glue code for specialized services such as security and fail-over recovery.
- Relational database management systems.
- COTS applications for systems management, e.g., software distribution and remote database administration.

Conventional and Unconventional Wisdom

Assumptions at the beginning of the development cycle were that the use of infrastructure COTS products would provide the following benefits.

- Using COTS products would reduce development costs and overall schedule.
- As a corollary, the development cycle would be accelerated.
- Feasibility demonstrations could be put together quickly.
- End-product quality would be higher as measured by a richer feature set and increased system robustness (assuming the selected COTS product is mature) [1].
- COTS vendors would provide maintenance for their COTS products.

The experience of integrating infrastructure COTS products and developed code refocused attention and revealed an additional set of assumptions for future developments.

- Accelerated development catapults you immediately into an integration and test activity.
- Hands-on evaluation requires early simulated applications in an integrated environment; these simulated applications and other test software can represent significant development costs.
- Maintenance on identified problems is provided by the COTS software vendor, but problem investigation and identification by the integrator are the most costly parts of COTS software maintenance.
- Maintenance turnaround time by the vendors can be a significant problem.

Lifecycle Implications

The development method must be specifically tailored to accommodate COTS product integration. This entails a set of assumptions and constraints quite different from custom-built development. Some of the more important of these follow. For a description of the referenced development stages (definition and analysis, functional design, physical design, and construction and test), see Part I of this article.

The front-end processes in the definition and analysis stage must support concurrent requirements and COTS product analysis. The analysis prototype

in the functional design stage must provide for iteration and a flexible linkage between the COTS product evaluations and the feedback loop to requirements analysis.

During the construction stage, the development processes acquire a dual nature when COTS product integration is introduced. One process path is valid for COTS product integration, and another process path is valid for developing the glue code and custom-built components. These two process paths are equivalent but consist of different activities and products. In addition, all COTS products, glue code, and custom-built components must be integrated together to complete development.

During the construction stage, the development of glue code that integrates COTS products and fills in missing functionality is similar to the development of traditional software; the traditional process of coding, unit testing, and integration is applicable.

For COTS products, the construction stage is when COTS products undergo detailed tuning and configuration and when the interfaces and threads between components are exercised in a multi-COTS product environment. COTS product tuning, configuration, and integration have an analog to code and unit-test activities. Unit test with COTS products is "black-box" (vs. "white-box") testing, and the focus is on interfaces and COTS product behavior. For example, unit testing of the transaction processing monitor consisted of exercising all the application programming interface (API) calls supported by the product as configured within the target environment.

Traditional software maintenance activities must be expanded in scope and extended to provide continuing COTS product support. This support starts early in the lifecycle. Application developers must have early deliveries and training for partially completed infrastructure functionality to keep their development lifecycle within reasonable time frames. Developers also require onsite, hands-on direct support from infrastructure developers and integrators to ensure acceptance and proper use of the infrastructure products.

Configuration control must be organized and in place early to accommodate multiple versions of the COTS products and configuration files. Separate environments for development and integration must be well-defined and structured to accept the delivered COTS products. Early support for multiple baselines must be in place as the combinations of COTS products become complex.

Throughout the lifecycle, feedback loops allow ongoing re-evaluation of the COTS products. Analysis prototypes (functional design stage) determine feasibility of a COTS-based solution and provide feedback to the requirements definition (definition and analysis stage). Design prototypes (physical design stage) provide hands-on experience with potential COTS products and feedback to the COTS product selection process (functional design stage). Detailed design prototypes (physical design stage) exercise functionality of selected COTS products, verify adherence and consistency with design expectations, reveal detailed behavior and performance characteristics, and give insight into the invocation parameters. The demonstration prototype (construction and test stages) is used to unit test the COTS products using black-box testing to simulate application behavior or environment. Each stage is a potential source of feedback to previous stages.

Practical Considerations

The following practical considerations were encountered during two years of experience using the IIDA.

- The COTS product integrator does not develop the COTS product but still must internally know it. The integrator must understand the complete set of capabilities provided by the COTS product to select the appropriate subset of capabilities based on application developer needs for a given release of infrastructure. The integrator must understand the limitations and nuances of the COTS product to exercise it. For example, does it run on all of the required platforms? Does it operate the way it is intended? Does it have a heritage from a different paradigm (PC vs. UNIX workstation)?
- The system administrators and configuration management staff need to know how to configure the COTS products. Few complex COTS products work straight out of the box. To support early prototypes and evaluations, not only do the designers and developers need to understand the products, the development system administrators need to understand how to install and manage the product configuration. In addition, configuration management needs to understand how to configure the product versions.
- "COTS castles are often built on the sand of configuration files." Configuration files and data can be as complex as code. They must be understood. For example, a transaction processing monitor configuration file is inherently complex; training is required to know how to use it. Configuration files can be site-specific and require a strategy to manage files for different sites including site-specific parameters, implementation requests, and file distribution.
- When installing infrastructure components in new sites, the following documents that are not part of normal lifecycles are critical for the configuration of COTS products.
  - Release notes (installation guidelines, operational parameters, tuning guidelines, etc.)
  - Site configuration guidelines (guidelines to help site designers choose appropriate hardware and software suites and rules for scaling and resource allocation).
- Version compatibility between COTS products, the operating system, and glue code is critical. This also applies to different sites including the external integration and test function. Software problems and nuances of use discovered during integration are not necessarily embedded in selected COTS products

but often derive from specific characteristics of operating system versions or communications protocols. If application developers, infrastructure developers, and test sites are allowed to independently manage their computing platform configurations (including operating system and database management system), trouble-shooting infrastructure anomalies is extremely difficult.
- Licensing adds a dimension of complexity and needs to be worked with early. Issues include the number and types of licenses required for the environment. Short-term COTS evaluation licenses need to be managed, and transition needs to be planned from evaluation to product license. Procurement of production licenses within government agencies can require a long lead time and needs to start early with the Bill of Materials (BOM).

Technical Management Considerations

The following considerations can be easily overlooked during the planning cycle.

- The development facility including hardware, development tools, and configuration management must be ready to go before the first COTS product arrives for prototyping. Facility readiness fuels the accelerated development that using COTS products can provide but moves the requirement for a fully implemented development facility to early in the effort. Determining COTS suitability requires a realistic target configuration with a strong system administration team in place from the start.
- The BOM represents the contract for COTS products and versions. It is required early for field development sites and is essential for successful deployment.
- Technology infusion occurs by virtue of COTS product upgrades whether it is planned or not. Product upgrades can occur during any phase of the lifecycle. Allowing for technology infusion can exploit new potential products on the market.
- The investment in training is a significant but often overlooked cost of using COTS products. Management needs to plan for the expertise of individuals to be shared across organizations. In particular, field sites need training, especially in system administration.

Conclusion

Integration with COTS software products requires adjustment and accommodations to the development approach vs. traditional software development. Preparations must be made to start prototyping activities and integration activities immediately to exploit COTS product advantages and accelerate development. Additional resources must be allocated for late in the development cycle to provide maintenance and support to the user community, i.e., the application developers. ◆

Acknowledgment

We thank David P. Maloney, a software development manager at TRW, for his contributions to this article. Many of the insights in the application of IIDA resulted from his work.

About the Authors

Greg Fox is a TRW Systems Integration Group technical fellow and the director of technology for the Information Services Division. He has 28 years experience in mostly large or complex information systems. He has led the architecture development and system integration for several large COTS-based systems and has been TRW's information systems infrastructure project manager and chief architect for the Integration Support Contract for Internal Revenue Service (IRS) modernization. He has engineering degrees from Massachusetts Institute of Technology and University of Southern California and has published over a dozen papers.

TRW, Inc.
MVA1/4943
12900 Federal Systems Park Drive
Fairfax, VA 22033
Voice: 703-876-4396
E-mail: greg.fox@trw.com

Steven Marcom is a senior systems analyst with the TRW Information Services Division. He has 30 years managerial and technical experience developing computer systems for civil government, defense, and commercial customers. He was TRW's systems lifecycle deputy manager and information systems infrastructure process engineer for the Integration Support Contract for IRS modernization. He has been active in Rapid Application Development, COTS integration, and prototyping activities. He has a bachelor's degree from Pomona College and a master's degree from the American University of Beirut, both in mathematics. He teaches software development and integration at TRW.

TRW, Inc.
FP1
12900 Federal Systems Park Drive
Fairfax, VA 22033
Voice: 703-803-4814
E-mail: marcoms@gisdbbs.gisd.trw.com

Karen W. Lantner is a program/project manager for EDS in New York City. She has 24 years management and technical experience, during which she has managed and consulted on large federal software development and COTS integration projects. A member of the team that developed the EDS Systems Life Cycle Methodology, she continues to have a special interest in software development methods. She has a bachelor's degree and a master's degree from Brown University.

EDS
A5N-B50
13600 EDS Drive
Herndon, VA 22071
Voice: 800-336-4498, box no. 52032
E-mail: karen.w.lantner@aexp.com

Reference

1. Langley, R.J., "COTS Integration Issues, Risks, and Approaches," *SIG Technology Review*, TRW Systems and Integration Group, Vol. 2, No. 2, Winter 1994, pp. 4-14.